Semi-Implicit Update

1 Derivation

At this point in the simulation, we want to update the grid node velocity. Explicitly, this is just

$$\boldsymbol{v}_{\boldsymbol{i}}^{n+1} = \boldsymbol{v}_{\boldsymbol{i}}^{n} + \frac{\partial \boldsymbol{v}_{\boldsymbol{i}}^{n}}{\partial n} = \boldsymbol{v}_{\boldsymbol{i}}^{n} + \Delta t \frac{\partial \boldsymbol{v}_{\boldsymbol{i}}^{n}}{\partial t} = \boldsymbol{v}_{\boldsymbol{i}}^{n} + \Delta t \frac{\boldsymbol{f}_{\boldsymbol{i}}^{n}}{m_{\boldsymbol{i}}}, \tag{1}$$

by Newton's second law. The backward Euler method can be used to express the new velocity implicitly:

$$\boldsymbol{v}_{\boldsymbol{i}}^{n+1} = \boldsymbol{v}_{\boldsymbol{i}}^{n} + \frac{\partial \boldsymbol{v}_{\boldsymbol{i}}^{n+1}}{\partial n} = \boldsymbol{v}_{\boldsymbol{i}}^{n} + \Delta t \frac{\partial \boldsymbol{v}_{\boldsymbol{i}}^{n+1}}{\partial t} = \boldsymbol{v}_{\boldsymbol{i}}^{n} + \Delta t \frac{\boldsymbol{f}_{\boldsymbol{i}}^{n+1}}{m_{\boldsymbol{i}}}. \tag{2}$$

The "semi-implicit" update employed by the authors essentially interpolates between these two schemes:

$$\begin{aligned}
\boldsymbol{v}_{\boldsymbol{i}}^{n+1} &= \boldsymbol{v}_{\boldsymbol{i}}^{n} + (1-\beta)\Delta t \frac{\boldsymbol{f}_{\boldsymbol{i}}^{n}}{m_{\boldsymbol{i}}} + \beta \Delta t \frac{\boldsymbol{f}_{\boldsymbol{i}}^{n+1}}{m_{\boldsymbol{i}}} \\
&= \boldsymbol{v}_{\boldsymbol{i}}^{n} + \frac{\Delta t}{m_{\boldsymbol{i}}} \Big(\boldsymbol{f}_{\boldsymbol{i}}^{n} + \beta \left(\boldsymbol{f}_{\boldsymbol{i}}^{n+1} - \boldsymbol{f}_{\boldsymbol{i}}^{n} \right) \Big) \\
&\approx \boldsymbol{v}_{\boldsymbol{i}}^{n} + \frac{\Delta t}{m_{\boldsymbol{i}}} \Big(\boldsymbol{f}_{\boldsymbol{i}}^{n} + \beta \Delta t \frac{\partial \boldsymbol{f}_{\boldsymbol{i}}^{n}}{\partial t} \Big).
\end{aligned}$$

Note that

$$\frac{\partial \boldsymbol{f}_{\boldsymbol{i}}^{n}}{\partial t} \approx \sum_{\boldsymbol{j}} \frac{\partial \boldsymbol{f}_{\boldsymbol{i}}^{n}}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} \boldsymbol{v}_{\boldsymbol{j}}^{n+1},$$

where $\hat{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{x}_{\boldsymbol{i}} + \Delta t\, \boldsymbol{v}_{\boldsymbol{i}}$. So, the update scheme is

$$\boldsymbol{v}_{\boldsymbol{i}}^{n+1} = \boldsymbol{v}_{\boldsymbol{i}}^{n} + \frac{\Delta t}{m_{\boldsymbol{i}}} \left(\boldsymbol{f}_{\boldsymbol{i}}^{n} + \beta \Delta t \sum_{\boldsymbol{j}} \frac{\partial \boldsymbol{f}_{\boldsymbol{i}}^{n}}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} \boldsymbol{v}_{\boldsymbol{j}}^{n+1} \right). \tag{3}$$

You can see that when $\beta = 0$, this update is equivalent to the explicit update, and when $\beta = 1$, this update is (approximately) equivalent to the backward Euler update.

Define the explicitly updated velocity as

$$\boldsymbol{v}_{\boldsymbol{i}}^{*} = \boldsymbol{v}_{\boldsymbol{i}}^{n} + \Delta t \frac{\boldsymbol{f}_{\boldsymbol{i}}^{n}}{m_{\boldsymbol{i}}} \tag{4}$$

By definition,

$$\frac{\partial^2 \Phi^n}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}} \partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} = -\frac{\partial \boldsymbol{f}_{\boldsymbol{i}}^{n}}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}}. \tag{5}$$

Using (3), (4), and (5), we have that

$$\boldsymbol{v}_{\boldsymbol{i}}^{*} = \boldsymbol{v}_{\boldsymbol{i}}^{n+1} + \frac{\beta \Delta t^2}{m_{\boldsymbol{i}}} \sum_{\boldsymbol{j}} \frac{\partial^2 \Phi^n}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}} \partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} \boldsymbol{v}_{\boldsymbol{j}}^{n+1} = \sum_{\boldsymbol{j}} \left(\boldsymbol{I} \delta_{\boldsymbol{ij}} + \frac{\beta \Delta t^2}{m_{\boldsymbol{i}}} \frac{\partial^2 \Phi^n}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}} \partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} \right) \boldsymbol{v}_{\boldsymbol{j}}^{n+1}. \tag{6}$$

2 Implementation

This update can be formulated as a large system of linear equations. Define $\boldsymbol{V}$ as the vector of all the updated grid node velocities, such that

$$\boldsymbol{v} \triangleq \left(\boldsymbol{v}_{1x}^{n+1},\ \boldsymbol{v}_{1y}^{n+1},\ \boldsymbol{v}_{1z}^{n+1},\ \ldots\ ,\ \boldsymbol{v}_{gx}^{n+1},\ \boldsymbol{v}_{gy}^{n+1},\ \boldsymbol{v}_{gz}^{n+1}\right)^T,$$

where g is the total number of grid nodes. Likewise, define $\boldsymbol{V}^*$ as the vector of all the explicitly update grid node velocities, such that

$$\boldsymbol{v}^* \triangleq \left(\boldsymbol{v}_{1x}^*,\ \boldsymbol{v}_{1y}^*,\ \boldsymbol{v}_{1z}^*,\ \ldots\ ,\ \boldsymbol{v}_{gx}^*,\ \boldsymbol{v}_{gy}^*,\ \boldsymbol{v}_{gz}^*\right)^T.$$

We want to express (6) in matrix form as $\boldsymbol{v}^* = \boldsymbol{E}\boldsymbol{v}$. Both $\boldsymbol{v}$ and $\boldsymbol{v}^*$ have dimensions $3g \times 1$, so $\boldsymbol{E}$ is a matrix of dimensions $3g \times 3g$. The conjugate residual method is used to solve this large linear system.

2.1 Conjugate Residual Method

The conjugate residual method is an iterative method used to solve linear systems of the general form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. We'll use $\boldsymbol{v}^*$ as our initial guess $\boldsymbol{v}_0$. The algorithm proceeds as follows:

$\boldsymbol{r}_0 = \boldsymbol{v}^* - \boldsymbol{E}\boldsymbol{v}_0$
$\boldsymbol{s}_0 = \boldsymbol{E}\boldsymbol{r}_0$
$\boldsymbol{p}_0 = \boldsymbol{r}_0$
$\boldsymbol{q}_0 = \boldsymbol{s}_0$
$\gamma_0 = \boldsymbol{r}_0^T\boldsymbol{s}_0$
for $k = 0, 1, 2, \ldots$ **do**
  $\alpha_k = \dfrac{\gamma_k}{\boldsymbol{q}_k^T\boldsymbol{q}_k}$
  $\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \alpha_k\boldsymbol{p}_k$
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k\boldsymbol{q}_k$
  $\boldsymbol{s}_{k+1} = \boldsymbol{E}\boldsymbol{r}_{k+1}$
  $\beta_k = \dfrac{\boldsymbol{r}_{k+1}^T\boldsymbol{s}_{k+1}}{\gamma_k}$
  $\gamma_{k+1} = \beta_k\gamma_k$
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k\boldsymbol{p}_k$
  $\boldsymbol{q}_{k+1} = \boldsymbol{s}_{k+1} + \beta_k\boldsymbol{q}_k$
end for

This will continue until some stopping criterion is satisfied; presumably, when $\|\alpha_k\boldsymbol{p}_k\|^2$ falls below some threshold. The important thing to note is that we don't actually have to explicitly define the matrix $\boldsymbol{E}$, we just need to be able to compute $\boldsymbol{E}\boldsymbol{u}$ given some vector $\boldsymbol{u}$.

2.2 Computation of $\boldsymbol{E}\boldsymbol{u}$

The key term in (6) is the Hessian of the potential. The "action" of this Hessian on an arbitrary increment $\delta\boldsymbol{u}$ is

$$\delta\boldsymbol{f}_{\boldsymbol{i}} = -\sum_{\boldsymbol{j}} \frac{\partial^2\Phi}{\partial\hat{\boldsymbol{x}}_{\boldsymbol{i}}\partial\hat{\boldsymbol{x}}_{\boldsymbol{j}}}\delta\boldsymbol{u}_{\boldsymbol{j}} = -\sum_p V_p^0\boldsymbol{A}_p(\boldsymbol{F}_{Ep}^n)^T\nabla w_{\boldsymbol{j}p}^n, \tag{7}$$

where

$$\boldsymbol{A}_p = \frac{\partial^2\Psi}{\partial\boldsymbol{F}_{Ep}\partial\boldsymbol{F}_{Ep}} : \left(\sum_{\boldsymbol{j}}\delta\boldsymbol{u}_{\boldsymbol{j}}(\nabla w_{\boldsymbol{j}p})^T\boldsymbol{F}_{Ep}^n\right). \tag{8}$$

Generally, the notation $a = \boldsymbol{B} : \boldsymbol{C}$ indicates the Frobenius inner product of matrices B and C, where

$$a = \boldsymbol{B} : \boldsymbol{C} = \sum_i \sum_j \boldsymbol{B}_{ij} \boldsymbol{C}_{ij}.$$

The authors extend this notation to the form seen in (8), $\boldsymbol{A} = \boldsymbol{B} : \boldsymbol{C}$ to indicate that

$$\boldsymbol{A}_{ij} = \boldsymbol{B}_{ij} : \boldsymbol{C}. \tag{9}$$

In other words, $\boldsymbol{B}$ is some matrix of sub-matrices, such that $\boldsymbol{B}_{ij}$ itself is a matrix. Each entry of $\boldsymbol{A}$ is the Frobenius inner product of the corresponding sub-matrix in $\boldsymbol{B}$ with $\boldsymbol{C}$. In the case of (8), $\boldsymbol{A}$ and $\boldsymbol{C}$ are 3×3 matrices while $\boldsymbol{B}$ is a 3×3 matrix of 3×3 sub-matrices.

Let's define $\boldsymbol{u}^* \triangleq \boldsymbol{E}\boldsymbol{u}$. Equation (6) can be rewritten into the form of (7) and (8)

$$\boldsymbol{u}^*(\boldsymbol{u}) = \sum_{\boldsymbol{j}} \left(\boldsymbol{I}\frac{\delta_{ij}}{\Delta t} + \frac{\beta \Delta t}{m_{\boldsymbol{i}}} \frac{\partial^2 \Phi^n}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}} \partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} \right) \Delta t \boldsymbol{u}_{\boldsymbol{j}} = \boldsymbol{u}_{\boldsymbol{i}} + \frac{\beta \Delta t}{m_{\boldsymbol{i}}} \sum_{\boldsymbol{j}} \frac{\partial^2 \Phi^n}{\partial \hat{\boldsymbol{x}}_{\boldsymbol{i}} \partial \hat{\boldsymbol{x}}_{\boldsymbol{j}}} (\Delta t \boldsymbol{u}_{\boldsymbol{j}}), \tag{10}$$

where $\Delta t \boldsymbol{u}_{\boldsymbol{j}}$ is equivalent to $\delta \boldsymbol{u}_{\boldsymbol{j}}$. Let's define

$$\delta \boldsymbol{F}_{Ep} \triangleq \sum_{\boldsymbol{j}} \Delta t \boldsymbol{u}_{\boldsymbol{j}} (\nabla w_{\boldsymbol{j}p})^T \boldsymbol{F}^n_{Ep},$$

given this vector $\boldsymbol{u}_{\boldsymbol{j}}$, such that

$$\boldsymbol{A}_p = \frac{\partial^2 \Psi}{\partial \boldsymbol{F}_{Ep} \partial \boldsymbol{F}_{Ep}} : \delta \boldsymbol{F}_{Ep}.$$

In the technical report, the authors show that

$$\frac{\partial^2 \Psi}{\partial \boldsymbol{F}_{Ep} \partial \boldsymbol{F}_{Ep}} : \delta \boldsymbol{F}_{Ep} = 2\mu \, \delta \boldsymbol{F}_{Ep} - 2\mu \, \delta \boldsymbol{R}_{Ep} + \lambda J_{Ep} \boldsymbol{F}^{-T}_{Ep} (J_{Ep} \boldsymbol{F}^{-T}_{Ep} : \delta \boldsymbol{F}_{Ep}) + \lambda (J_{Ep} - 1) \, \delta (J_{Ep} \boldsymbol{F}^{-T}_{Ep}), \tag{11}$$

where as usual, $\boldsymbol{F}_{Ep} = \boldsymbol{R}_{Ep} \boldsymbol{S}_{Ep}$ by polar decomposition, and $J_{Ep} = \det(\boldsymbol{F}_{Ep})$.

The matrix $J_{Ep} \boldsymbol{F}^{-T}_{Ep}$ is the cofactor matrix of $\boldsymbol{F}_{Ep}$, so its entries can be expressed as polynomials in the entries of $\boldsymbol{F}_{Ep}$. So the 3×3 matrix of 3×3 sub-matrices $\frac{\partial}{\partial \boldsymbol{F}_{Ep}} (J_{Ep} \boldsymbol{F}^{-T}_{Ep})$ can be expressed in terms of the entries of $\boldsymbol{F}_{Ep}$, which allows us to compute

$$\delta (J_{Ep} \boldsymbol{F}^{-T}_{Ep}) = \frac{\partial}{\partial \boldsymbol{F}_{Ep}} (J_{Ep} \boldsymbol{F}^{-T}_{Ep}) : \delta \boldsymbol{F}_{Ep}.$$

We also have to compute $\delta \boldsymbol{R}_{Ep}$:

$$\begin{gathered} \delta \boldsymbol{F}_{Ep} = \delta \boldsymbol{R}_{Ep} \boldsymbol{S}_{Ep} + \boldsymbol{R}_{Ep} \delta \boldsymbol{S}_{Ep} \\ \Rightarrow \boldsymbol{R}^T_{Ep} \delta \boldsymbol{F}_{Ep} - \delta \boldsymbol{F}^T_{Ep} \boldsymbol{R}_{Ep} = (\boldsymbol{R}^T_{Ep} \delta \boldsymbol{R}_{Ep}) \boldsymbol{S}_{Ep} + \boldsymbol{S}_{Ep} (\boldsymbol{R}^T_{Ep} \delta \boldsymbol{R}_{Ep}). \end{gathered} \tag{12}$$

Equation (12) might not seem especially helpful, but we can take advantage of the fact that $\boldsymbol{R}^T_{Ep} \delta \boldsymbol{R}_{Ep}$ is skew symmetric, so it only has three independent entries. In fact, the left-hand side of (12) is also skew-symmetric. Let's define

$$\boldsymbol{V} \triangleq \boldsymbol{R}^T_{Ep} \delta \boldsymbol{F}_{Ep} - \delta \boldsymbol{F}^T_{Ep} \boldsymbol{R}_{Ep} = \begin{pmatrix} 0 & d & e \\ -d & 0 & f \\ -e & -f & 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{U} \triangleq \boldsymbol{R}^T_{Ep} \delta \boldsymbol{R}_{Ep} = \begin{pmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{pmatrix}.$$

We have the values of $\boldsymbol{V}$ and $\boldsymbol{S}_{Ep} = \{s_{ij}\}$. The three independent values of $\boldsymbol{U}$ can be computed directly by solving the 3×3 system

$$\begin{pmatrix} s_{00} + s_{11} & s_{21} & -s_{02} \\ s_{12} & s_{00} + s_{22} & s_{01} \\ -s_{02} & s_{10} & s_{11} + s_{22} \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} d \\ e \\ f \end{pmatrix}$$

Then, $\delta \boldsymbol{R}_{Ep} = \boldsymbol{R}_{Ep} (\boldsymbol{R}^T_{Ep} \delta \boldsymbol{R}_{Ep})$.